

Appendix A

ADDLW	ADD literal to W								
Syntax:	[label]: ADDLW k								
Operands:	0 ≤ k ≤ 255								
Operation:	(W) + k → W								
Status Affected:	N,OV, C, DC, Z								
Encoding:	0000 1111 kkkk kkkk								
Description:	The contents of W are added to the 8-bit literal 'k' and the result is placed in W.								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write to W						

Example: ADDLW 0x15

Before Instruction

W = 0x10

After Instruction

W = 0x25

ADDWF	ADD W to f								
Syntax:	[label] ADDWF f,d,a								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(W) + (f) → dest								
Status Affected:	N,OV, C, DC, Z								
Encoding:	0010 01da ffff ffff								
Description:	Add W to register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: ADDWF REG, 0, 0

Before Instruction

W = 0x17
REG = 0xC2

After Instruction

W = 0xD9
REG = 0xC2

ADDWFC	ADD W and Carry bit to f		
Syntax:	[label] ADDWFC f,d,a		
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$		
Operation:	$(W) + (f) + (C) \rightarrow \text{dest}$		
Status Affected:	N,OV, C, DC, Z		
Encoding:	0010 00da ffff ffff		
Description:	Add W, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden.		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write to destination

Example: ADDWFC REG, 0, 1

Before Instruction

Carry bit = 1
REG = 0x02
W = 0x4D

After Instruction

Carry bit = 0
REG = 0x02
W = 0x50

ANDLW	AND literal with W		
Syntax:	[label] ANDLW k		
Operands:	$0 \leq k \leq 255$		
Operation:	(W) .AND. k \rightarrow W		
Status Affected:	N,Z		
Encoding:	0000 1011 kkkk kkkk		
Description:	The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: ANDLW 0x5F

Before Instruction

W = 0xA3

After Instruction

W = 0x03

ANDWF	AND W with f								
Syntax:	[label] ANDWF f,d,a								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(W) .AND. (f) → dest								
Status Affected:	N,Z								
Encoding:	0001 01da ffff ffff								
Description:	The contents of W are AND'ed with register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: ANDWF REG, 0, 0

Before Instruction

W = 0x17
REG = 0xC2

After Instruction

W = 0x02
REG = 0xC2

BC	Branch if Carry												
Syntax:	[label] BC n												
Operands:	-128 ≤ n ≤ 127												
Operation:	if carry bit is '1' (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	1110 0010 nnnn nnnn												
Description:	If the Carry bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:													
If Jump:													
	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:													
	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	No operation										

Example: HERE BC 5

Before Instruction

PC = address (HERE)

After Instruction

If Carry PC = 1; address (HERE+12)
If Carry PC = 0; address (HERE+2)

BCF		Bit Clear f	
Syntax:	[label] BCF f,b,a		
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]		
Operation:	0 → f		
Status Affected:	None		
Encoding:	1001 bbba ffff ffff		
Description:	Bit 'b' in register 'f' is cleared. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write register f

Example: BCF FLAG_REG, 7, 0

Before Instruction
FLAG_REG = 0xC7

After Instruction
FLAG_REG = 0x47

BN		Branch if Negative	
Syntax:	[label] BN n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if negative bit is '1' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0110 nnnn nnnn		
Description:	If the Negative bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

If No Jump:

Q1	Q2	Q3	Q4

Example: HERE BN Jump

Before Instruction
PC = address (HERE)

After Instruction
If Negative = 1;
PC = address (Jump)
If Negative = 0;
PC = address (HERE+2)

BNC		Branch if Not Carry	
Syntax:	[label] BNC n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if carry bit is '0' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0011 nnnn nnnn		
Description:	If the Carry bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:			

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNC Jump

Before Instruction	
PC	= address (HERE)
After Instruction	
If Carry	= 0;
PC	= address (Jump)
If Carry	= 1;
PC	= address (HERE+2)

BNN		Branch if Not Negative	
Syntax:	[label] BNN n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if negative bit is '0' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0111 nnnn nnnn		
Description:	If the Negative bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:			

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNN Jump

Before Instruction	
PC	= address (HERE)
After Instruction	
If Negative	= 0;
PC	= address (Jump)
If Negative	= 1;
PC	= address (HERE+2)

INV	Branch if Not Overflow			
Syntax:	[label] BNV n			
Operands:	-128 ≤ n ≤ 127			
Operation:	if overflow bit is '0' (PC) + 2 + 2n → PC			
Status Affected:	None			
Encoding:	1110	0101	nnnn	nnnn
Description:	<p>If the Overflow bit is '0', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.</p>			
Words:	1			
Cycles:	1(2)			
Q Cycle Activity:				
If Jump:	Q1	Q2	Q3	Q4
	Decode	Read literal 'n'	Process Data	Write to PC
	No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decod	Read literal 'n'	Process Data	No operation

Example: HERE BNV Jump

Before Instruction
 PC = address (HERE)

After Instruction
 If Overflow = 0;
 PC = address (Jump)
 If Overflow = 1;
 PC = address (HERE+2)

BNZ	Branch if Not Zero			
Syntax:	[label] BNZ n			
Operands:	-128 ≤ n ≤ 127			
Operation:	if zero bit is '0' (PC) + 2 + 2n → PC			
Status Affected:	None			
Encoding:	1110	0001	nnnn	nnnn
Description:	<p>If the Zero bit is '0', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.</p>			
Words:	1			
Cycles:	1(2)			
Q Cycle Activity:				
If Jump:	Q1	Q2	Q3	Q4
	Decode	Read literal 'n'	Process Data	Write to PC
	No operation	No operation	No operation	No operation
If No Jump:	Q1	Q2	Q3	Q4
	Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNZ Jump

Before Instruction
 PC = address (HERE)

After Instruction
 If Zero = 0;
 PC = address (Jump)
 If Zero = 1;
 PC = address (HERE+2)

BRA Unconditional Branch													
Syntax:	[label] BRA n												
Operands:	-1024 ≤ n ≤ 1023												
Operation:	(PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1101</td><td>0nnn</td><td>nnnn</td><td>nnnn</td></tr></table>	1101	0nnn	nnnn	nnnn								
1101	0nnn	nnnn	nnnn										
Description:	Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table style="width: 100%; border-collapse: collapse;"><thead><tr><th style="text-align: center; width: 25%;">Q1</th><th style="text-align: center; width: 25%;">Q2</th><th style="text-align: center; width: 25%;">Q3</th><th style="text-align: center; width: 25%;">Q4</th></tr></thead><tbody><tr><td style="text-align: center;">Decode</td><td style="text-align: center;">Read literal 'n'</td><td style="text-align: center;">Process Data</td><td style="text-align: center;">Write to PC</td></tr><tr><td style="text-align: center;">No operation</td><td style="text-align: center;">No operation</td><td style="text-align: center;">No operation</td><td style="text-align: center;">No operation</td></tr></tbody></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										

Example: HERE BRA Jump

Before Instruction PC = address (HERE)

After Instruction PC = address (Jump)

BSF Bit Set f									
Syntax:	[label] BSF f,b,a								
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ {0,1}								
Operation:	1 → f								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1000</td><td>bbba</td><td>ffff</td><td>ffff</td></tr></table>	1000	bbba	ffff	ffff				
1000	bbba	ffff	ffff						
Description:	Bit 'b' in register 'f' is set. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table style="width: 100%; border-collapse: collapse;"><thead><tr><th style="text-align: center; width: 25%;">Q1</th><th style="text-align: center; width: 25%;">Q2</th><th style="text-align: center; width: 25%;">Q3</th><th style="text-align: center; width: 25%;">Q4</th></tr></thead><tbody><tr><td style="text-align: center;">Decode</td><td style="text-align: center;">Read register f</td><td style="text-align: center;">Process Data</td><td style="text-align: center;">Write register f</td></tr></tbody></table>	Q1	Q2	Q3	Q4	Decode	Read register f	Process Data	Write register f
Q1	Q2	Q3	Q4						
Decode	Read register f	Process Data	Write register f						

Example: BSF FLAG_REG, 7, 1

Before Instruction FLAG_REG= 0x0A

After Instruction FLAG_REG= 0x8A

BTFSC Bit Test File, Skip if Clear	
Syntax:	[label] BTFSC f,b,a
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ {0,1}
Operation:	skip if (f) = 0
Status Affected:	None
Encoding:	1011 bbba ffff ffff
Description:	If bit 'b' in register 'f' is 0 then the next instruction is skipped. If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE BTFSC FLAG, 1, 0
FALSE :
TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

BTFSS Bit Test Fil , Skip if Set	
Syntax:	[label] BTFSS f,b,a
Op rands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ {0,1}
Operation:	skip if (f) = 1
Status Affected:	None
Encoding:	1010 bbba ffff ffff
Description:	If bit 'b' in register 'f' is 1 then the next instruction is skipped. If bit 'b' is 1, then the next instruction fetched during the current instruction execution, is discarded and an NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE BTFSS FLAG, 1, 0
FALSE :
TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

BTG		Bit Toggle f	
Syntax:	[label] BTG f,b,a		
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]		
Operation:	(f) → f		
Status Affected:	None		
Encoding:	0111 bbba ffff ffff		
Description:	Bit 'b' in data memory location 'f' is inverted. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
	Q1 Q2 Q3 Q4		
	Decode Read register 'f'	Process Data	Write register 'f'

Example: BTG PORTC, 4, 0

Before Instruction:
PORTC = 0111 0101 [0x75]

After Instruction:
PORTC = 0110 0101 [0x65]

BV		Branch if Overflow	
Syntax:	[label] BV n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if overflow bit is '1' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0100 xxxx xxxx		
Description:	If the Overflow bit is '1', then the program will branch.		
		The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.	
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:	Q1 Q2 Q3 Q4		
	Decode Read literal 'n'	Process Data	Write to PC
	No operation	No operation	No operation
If No Jump:	Q1 Q2 Q3 Q4		
	Decode Read literal 'n'	Process Data	No operation

Example: HERE BV Jump

Before Instruction
PC = address (HERE)

After Instruction
If Overflow = 1;
PC = address (Jump)
If Overflow = 0;
PC = address (HERE+2)

BZ Branch if Zero			
Syntax:	[label] BZ n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if Zero bit is '1' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0000 nnnn nnnn		
Description:	If the Zero bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:			
Q1 Q2 Q3 Q4			
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1 Q2 Q3 Q4			
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BZ Jump

Before Instruction
PC = address (HERE)

After Instruction
If Zero = 1;
PC = address (Jump)
If Zero = 0;
PC = address (HERE+2)

CALL Subroutine Call			
Syntax:	[label] CALL k,s		
Operands:	0 ≤ k ≤ 1048575 s ∈ {0,1}		
Operation:	(PC) + 4 → TOS, k → PC<20:1>, if s = 1 (W) → WS, (STATUS) → STATUS, (BSR) → BSRS		
Status Affected:	None		
Encoding:	1110 110s k ₁₉ k ₁₈ k ₇ k ₆ k ₅ k ₄ k ₃ k ₂ k ₁ k ₀		
Description:	Subroutine call of entire 2M byte memory range. First, return address (PC+4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUS and BSRS. If 's' = 0, no update occurs (default). Then the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-cycle instruction.		
Words:	2		
Cycles:	2		
Q Cycle Activity:			
Q1 Q2 Q3 Q4			
Decode	Read literal 'k'<7:0>, Push PC to stack	Push PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: HERE CALL THERE, fast

Before Instruction
PC = Address (HERE)

After Instruction
PC = Address (THERE)
TOS = Address (HERE + 4)
WS = W
BSRS = BSR
STATUS = STATUS

CLRF	Clear f								
Syntax:	[label] CLRF f,a								
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$								
Operation:	$000h \rightarrow f$ $1 \rightarrow Z$								
Status Affected:	Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr> </table>	0110	101a	ffff	ffff				
0110	101a	ffff	ffff						
Description:	Clears the contents of the specified register. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register \ddagger</td><td>Process Data</td><td>Write register \ddagger</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register \ddagger	Process Data	Write register \ddagger
Q1	Q2	Q3	Q4						
Decode	Read register \ddagger	Process Data	Write register \ddagger						

Example: CLRF FLAG_REG,1

Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00

CLRWDT	Clear Watchdog Timer								
Syntax:	[label] CLRWDT								
Operands:	None								
Operation:	$000h \rightarrow WDT$, $000h \rightarrow WDT$ postscaler, $1 \rightarrow TO$, $1 \rightarrow PD$								
Status Affected:	TO, PD								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0100</td></tr> </table>	0000	0000	0000	0100				
0000	0000	0000	0100						
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits TO and PD are set.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>No operation</td><td>Process Data</td><td>No operation</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	No operation	Process Data	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	Process Data	No operation						

Example: CLRWDT

Before Instruction
WDT counter = ?
After Instruction
WDT counter = 0x00
WDT Postscaler = 0
TO = 1
PD = 1

COMF	Complement f								
Syntax:	[label] COMF f,d,a								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(f) \rightarrow \text{dest}$								
Status Affected:	N,Z								
Encoding:	0001 11da ffff ffff								
Description:	The contents of register f are complemented. If d is 0 the result is stored in W . If d is 1 the result is stored back in register f (default). If a is 0 Virtual bank will be selected, overriding the BSR value. If a = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register f</td> <td>Process Data</td> <td>Write to destination</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register f	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register f	Process Data	Write to destination						

CPFSEQ	Compare f with W, skip if f = W
Syntax:	[label] CPFSEQ f,a
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	$(f) - (W)$, skip if $(f) = (W)$ (unsigned comparison)
Status Affected:	None
Encoding:	0110 001a ffff ffff
Description:	Compares the contents of data memory location f to the contents of W by performing an unsigned subtraction. If $f = W$ then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If a is 0 Virtual bank will be selected, overriding the BSR value. If a = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2)
	Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	No operation
If skip:			
Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
If skip and followed by 2-word instruction:			
Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE CPFSEQ REG, 0
NEQUAL :
EQUAL :

Before Instruction
PC Address = HERE
W = ?
REG = ?

After Instruction

If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

CPFSGT	Compare f with W, skip if f > W
Syntax:	[label] CPFSGT f,a
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(f) – (W), skip if (f) > (W) (unsigned comparison)
Status Affected:	None
Encoding:	0110 010a ffff ffff
Description:	Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater than the contents of W then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE CPFSGT REG, 0
NGREATER :
GREATER :

Before Instruction

PC = Address (HERE)
W = ?

After Instruction

If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

CPFSLT

Compare f with W, skip if f < W

Syntax: [label] CPFSLT f,a

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding: 0110 000a ffff ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).

Words: 1

Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE CPFSLT REG, 1
NLESS :
LESS :

Before Instruction

PC = Address (HERE)
W = ?

After Instruction

If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

DAW		Decimal Adjust W Register								
Syntax:	[label] DAW									
Operands:	None									
Operation:	If $[W<3:0> > 9]$ or $[DC = 1]$ then $(W<3:0>) + 6 \rightarrow W<3:0>;$ else $(W<3:0>) \rightarrow W<3:0>;$ If $[W<7:4> > 9]$ or $[C = 1]$ then $(W<7:4>) + 6 \rightarrow W<7:4>;$ else $(W<7:4>) \rightarrow W<7:4>;$									
Status Affected:	C									
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0111</td></tr> </table>		0000	0000	0000	0111				
0000	0000	0000	0111							
Description:	DAW adjusts the eight bit value in W resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.									
Words:	1									
Cycles:	1									
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register W</td><td>Process Data</td><td>Write W</td></tr> </table>		Q1	Q2	Q3	Q4	Decode	Read register W	Process Data	Write W
Q1	Q2	Q3	Q4							
Decode	Read register W	Process Data	Write W							

Example 1: DAW

Before Instruction

W = 0xA5
 C = 0
 DC = 0

After Instruction

W = 0x05
 C = 1
 DC = 0

Example 2:

Before Instruction

W = 0xCE
 C = 0
 DC = 0

After Instruction

W = 0x34
 C = 1
 DC = 0

DECF		Decrement f									
Syntax:	[label] DECF f,d,a										
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$										
Operation:	$(f) - 1 \rightarrow \text{dest}$										
Status Affected:	C,DC,N,OV,Z										
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table>		0000	01da	ffff	ffff					
0000	01da	ffff	ffff								
Description:	Decrement register f. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register f (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).										
Words:	1										
Cycles:	1										
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register f</td><td>Process Data</td><td>Write to destination</td></tr> </table>		Q1	Q2	Q3	Q4	Decode	Read register f	Process Data	Write to destination	
Q1	Q2	Q3	Q4								
Decode	Read register f	Process Data	Write to destination								

Example: DECF CNT, 1, 0

Before Instruction

CNT = 0x01
 Z = 0

After Instruction

CNT = 0x00
 Z = 1

DECFSZ	Decrement f, skip if 0			
Syntax:	[label] DECFSZ f,d,a			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) - 1 → dest, skip if result = 0			
Status Affected:	None			
Encoding:	0010 11da ffff ffff			
Description:	<p>The contents of register 'f' are decremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default).</p> <p>If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).</p>			
Words:	1			
Cycles:	1(2) Note: 3 cycles if skip and followed by a 2-word instruction			

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE	DECFSZ	CNT, 1, 1
	GOTO	LOOP
CONTINUE		

Before Instruction

PC = Address (HERE)

After Instruction

CNT =	CNT - 1
If CNT = 0;	
PC =	Address (CONTINUE)
If CNT ≠ 0;	
PC =	Address (HERE+2)

DCFSNZ	Decrement f, skip if not 0			
Syntax:	[label] DCFSNZ f,d,a			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) - 1 → dest, skip if result ≠ 0			
Status Affected:	None			
Encoding:	0100 11da ffff ffff			
Description:	<p>The contents of register 'f' are decremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default).</p> <p>If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).</p>			
Words:	1			
Cycles:	1(2) Note: 3 cycles if skip and followed by a 2-word instruction			

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE	DCFSNZ	TEMP, 1, 0
ZERO	:	
NZERO	:	

Before Instruction

TEMP = ?

After Instruction

TEMP	=	TEMP - 1,
If TEMP	=	0;
PC	=	Address (ZERO)
If TEMP	≠	0;
PC	=	Address (NZERO)

GOTO		Unconditional Branch			
Syntax:		[<i>label</i>]. GOTO <i>k</i>			
Operands:		0 ≤ <i>k</i> ≤ 1048575			
Operation:		<i>k</i> → PC<20:1>			
Status Affected:		None			
Encoding:					
1st word (k<7:0>)		1110	1111	<i>k</i> ₇ <i>kkk</i>	<i>kkkkk</i> ₀
2nd word(k<19:8>)		1111	<i>k</i> ₁₉ <i>kkk</i>	<i>kkkk</i>	<i>kkkkk</i> ₈
Description:		GOTO allows an unconditional branch anywhere within entire 2M byte memory range. The 20-bit value 'K' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.			
Words:		2			
Cycles:		2			
Q Cycle Activity:					
Q1	Q2	Q3	Q4		
Decode	Read literal 'k<7:0>,'	No operation	Read literal 'k<19:8>,' Write to PC		
No operation	No operation	No operation	No operation		

HALT		Halt Processor			
Syntax:		[<i>label</i>]. HALT			
Operands:		None			
Operation:		Processor halts execution after HALT instruction			
Status Affected:		None			
Encoding:		0000	0000	0000	0001
Description:		While functioning in emulation mode, execution of the halt instruction will halt processor execution. Toggling the HALT pin or resetting (MCLR = 0) will bring the device out of halt. HALT instruction is not recognized in non-emulation modes.			
Words:		1			
Cycles:		1			
Q Cycle Activity:					
Q1	Q2	Q3	Q4		
Decode	No operation	No operation	No operation	HALT	

Example: GOTO THERE

After Instruction

PC = Address (THERE)

INCF		Increment f	
Syntax:	[label] INCF f,d,a		
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]		
Operation:	(f) + 1 → dest		
Status Affected:	C,DC,N,OV,Z		
Encoding:	0010 10da ffff ffff		
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
	Q1 Q2 Q3 Q4		
	Decode Read register f Process Data Write to destination		

Example: INCF CNT, 1, 0

Before Instruction

CNT	=	0xFF
Z	=	0
C	=	?
DC	=	?

After Instruction

CNT	=	0x00
Z	=	1
C	=	1
DC	=	1

INCFSZ		Increment f, skip if 0	
Syntax:	[label] INCFSZ f,d,a		
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]		
Operation:	(f) + 1 → dest, skip if result = 0		
Status Affected:	None		
Encoding:	0011 11da ffff ffff		
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
Words:	1		
Cycles:	1(2)		
	Note: 3 cycles if skip and followed by a 2-word instruction		

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INCFSZ CNT, 1, 0
NZERO :
ZERO :

Before Instruction

PC = Address (HERE)

After Instruction

CNT	=	CNT + 1
If CNT	=	0;
PC	=	Address (ZERO)
If CNT	≠	0;
PC	=	Address (NZERO)

INFSNZ	Increment f, skip if not 0				
Syntax:	<code>[label] INFSNZ f,d,a</code>				
Operands:	$0 \leq f \leq 255$ $d \in \{0,1\}$ $a \in \{0,1\}$				
Operation:	$(f) + 1 \rightarrow \text{dest}$, skip if result $\neq 0$				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0100</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0100	10da	ffff	ffff
0100	10da	ffff	ffff		
Description:	<p>The contents of register f are incremented. If $'d'$ is 0 the result is placed in W. If $'d'$ is 1 the result is placed back in register f (default).</p> <p>If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If $'a'$ is 0 Virtual bank will be selected, overriding the BSR value. If $'a' = 1$, then the bank will be selected as per the BSR value (default).</p>				
Words:	1				
Cycles:	1(2)				
	Note: 3 cycles if skip and followed by a 2-word instruction				

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INFSNZ REG, 1, 0
ZERO
NZERO

Before Instruction

PC = Address (HERE)

After Instruction

REG = REG + 1

If REG $\neq 0$:

PC = Address (NZERO)

If REG = 0:

PC = Address (ZERO)

IORLW	Inclusive OR literal with W				
Syntax:	<code>[label] IORLW k</code>				
Operands:	$0 \leq k \leq 255$				
Operation:	$(W) .OR. k \rightarrow W$				
Status Affected:	N,Z				
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0000</td><td>1001</td><td>kkkk</td><td>kkkk</td></tr></table>	0000	1001	kkkk	kkkk
0000	1001	kkkk	kkkk		
Description:	The contents of W are OR'ed with the eight bit literal 'k'. The result is placed in W .				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
Q1	Q2	Q3	Q4		
Decode	Read literal 'k'	Process Data	Write to W		

Example: IORLW 0x35

Before Instruction

W = 0x9A

After Instruction

W = 0xBF

IORWF	Inclusive OR W with f								
Syntax:	[label] IORWF f,d,a								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(W) .OR. (f) → dest								
Status Affected:	N,Z								
Encoding:	0001 00da ffff ffff								
Description:	Inclusive OR W with register 'f'. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: IORWF RESULT, 0, 1

Before Instruction
 RESULT = 0x13
 W = 0x91

After Instruction
 RESULT = 0x13
 W = 0x93

MOVF	M v f								
Syntax:	[label] MOVF f,d,a								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	f → dest								
Status Affected:	N,Z								
Encoding:	0101 00da ffff ffff								
Description:	The contents of register 'f' is moved to a destination dependent upon the status of 'd'. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write W</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write W
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write W						

Example: MOVF REG, 0, 0

Before Instruction
 REG = 0x22
 W = 0xFF

After Instruction
 REG = 0x22
 W = 0x22

MOVFF Move f to f

Syntax:	[label] MOVFF f _s ,f _d
Operands:	0 ≤ f _s ≤ 4095 0 ≤ f _d ≤ 4095
Operation:	(f _s) → f _d
Status Affected:	None
Encoding:	
1st word (source)	1100 ffff ffff ffff _s
2nd word (destin.)	1111 ffff ffff ffff _d
Description:	<p>The contents of source register f_s are moved to destination register f_d. Location of source f_s can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination f_d can also be anywhere from 000h to FFFh.</p> <p>Either source or destination can be W (a useful special situation).</p> <p>MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).</p>

The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f [¶] (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register f [¶] (dest)

Example: MOVFF REG1, REG2

Before Instruction

REG1 = 0x33
REG2 = 0x11

After Instruction

REG1 = 0x33,
REG2 = 0x33

MOVLB Move literal to low nibble in BSR

Syntax:	[label] MOVLB k		
Operands:	0 ≤ k ≤ 255		
Operation:	k → BSR		
Status Affected:	None		
Encoding:	0000 0001 kkkk kkkk		
Description:	The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR

Example: MOVLB 5

Before Instruction

BSR register = 0x02

After Instruction
BSR register = 0x05

LFSR Move literal to FSR									
Syntax:	[label] LFSR f,k								
Operands:	0 ≤ f ≤ 2 0 ≤ k ≤ 4095								
Operation:	k → FSRf								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>1110</td><td>00ff</td><td>k₁₁kkk</td></tr> <tr><td>1111</td><td>0000</td><td>k₇kkk</td><td>kkkk</td></tr> </table>	1110	1110	00ff	k ₁₁ kkk	1111	0000	k ₇ kkk	kkkk
1110	1110	00ff	k ₁₁ kkk						
1111	0000	k ₇ kkk	kkkk						
Description:	The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'								
Words:	2								
Cycles:	2								
Q Cycle Activity:									
Q1	Q2	Q3	Q4						
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH						
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL						

Example: LFSR 2, 0x3AB

After Instruction

FSR2H = 0x03
FSR2L = 0xAB

MOVLW Move literal to W					
Syntax:	[label] MOVLW k				
Operands:	0 ≤ k ≤ 255				
Operation:	k → W				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1110</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1110	kkkk	kkkk
0000	1110	kkkk	kkkk		
Description:	The eight bit literal 'k' is loaded into W.				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
Q1	Q2	Q3	Q4		
Decode	Read literal 'k'	Process Data	Write to W		

Example: MOVLW 0x5A

After Instruction

W = 0x5A

MOVWF	Move W to f
Syntax:	[label] MOVWF f,a
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	$(W) \rightarrow f$
Status Affected:	None
Encoding:	0110 111a ffff ffff
Description:	Move data from W to register 'f'. Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: MOVWF REG, 0

Before Instruction

W = 0x4F
REG = 0xFF

After Instruction

W = 0x4F
REG = 0x4F

MULLW	Multiply Literal with W
Syntax:	[label] MULLW k
Operands:	$0 \leq k \leq 255$
Operation:	$(W) \times k \rightarrow \text{PRODH:PRODL}$
Status Affected:	None
Encoding:	0000 1101 kkkk kkck
Description:	An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH:PRODL

Example: MULLW 0xC4

Before Instruction

W = 0xE2
PRODH = ?
PRODL = ?

After Instruction

W = 0xE2
PRODH = 0xAD
PRODL = 0x08

MULWF	Multiply W with f		
Syntax:	[label] MULWF f,a		
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$		
Operation:	$(W) \times (f) \rightarrow \text{PRODH:PRODL}$		
Status Affected:	None		
Encoding:	0000 001a ffff ffff		
Description:	An unsigned multiplication is carried out between the contents of W and the register file location f . The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and f are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write register f

Example: MULWF REG, 1

Before Instruction

W = 0xC4
REG = 0xB5
PRODH = ?
PRODL = ?

After Instruction

W = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94

NEGF	Negate f		
Syntax:	[label] NEGF f,a		
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$		
Operation:	$(f) + 1 \rightarrow f$		
Status Affected:	N,OV, C, DC, Z		
Encoding:	0110 110a ffff ffff		
Description:	Location f is negated using two's complement. The result is placed in the data memory location f . If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write register f

Example: NEGF REG, 1

Before Instruction

REG = 0011 1010 [0x3A]

After Instruction

REG = 1100 0110 [0xC6]

NOP				
N Operation				
Syntax:	[<i>label</i>] NOP			
Operands:	None			
Operation:	No operation			
Status Affected:	None			
Encoding:	0000 1111	0000 xxxx	0000 xxxx	0000 xxxx
Description:	No operation.			
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	No operation	No operation	No operation

Example:

None.

POP				
P p T p of Return Stack				
Syntax:	[<i>label</i>] POP			
Operands:	None			
Operation:	(TOS) → bit bucket			
Status Affected:	None			
Encoding:	0000	0000	0000	0110
Description:	<p>The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack.</p> <p>This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.</p>			
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	No operation	POP TOS value	No operation

Example: **POP**
 GOTO **NEW**

Before Instruction

TOS	=	0031A2h
Stack (1 level down)	=	014332h

After Instruction

TOS	=	014332h
PC	=	NEW

PUSH		Push Top of Return Stack							
Syntax: [label] PUSH									
Operands: None									
Operation: $(PC+2) \rightarrow TOS$									
Status Affected: None									
Encoding: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0000</td><td>0101</td></tr></table>		0000	0000	0000	0101				
0000	0000	0000	0101						
Description: The PC+2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows to implement a software stack by modifying TOS, and then push it onto the return stack.									
Words:	1								
Cycles:	1								
Q Cycle Activity:									
Q1	Q2	Q3	Q4						
Decode	PUSH PC+2 onto return stack	No operation	No operation						

Example: **PUSH**

Before Instruction	
TOS	= 00345Ah
PC	= 000124h
After Instruction	
PC	= 000126h
TOS	= 000126h
Stack (1 level down)	= 00345Ah

RCALL		Branch Subroutine							
Syntax: [label] RCALL n									
Operands: $-1024 \leq n \leq 1023$									
Operation: $(PC) + 2 \rightarrow TOS$, $(PC) + 2 + 2n \rightarrow PC$									
Status Affected: None									
Encoding: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1101</td><td>1nnn</td><td>nnnn</td><td>nnnn</td></tr></table>		1101	1nnn	nnnn	nnnn				
1101	1nnn	nnnn	nnnn						
Description: Subroutine call with a jump upto 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$. This instruction is a two-cycle instruction.									
Words:	1								
Cycles:	2								
Q Cycle Activity:									
Q1	Q2	Q3	Q4						
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC						
No operation	No operation	No operation	No operation						

Example: **HERE RCALL Jump**

Before Instruction	
PC	= Address (HERE)
After Instruction	
PC	= Address (Jump)
TOS	= Address (HERE+2)

RESET	Reset				
Syntax:	<code>[label] RESET</code>				
Operands:	None				
Operation:	Reset all registers and flags that are affected by a MCLR reset.				
Status Affected:	All				
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0000</td><td>0000</td><td>1111</td><td>1111</td></tr></table>	0000	0000	1111	1111
0000	0000	1111	1111		
Description:	This instruction provides a way to execute a MCLR reset in software.				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
Q1	Q2	Q3	Q4		
Decode	Start reset	No operation	No operation		

Example: `RESET`

After Instruction

Registers = Reset Value
Flags* = Reset Value

RETFIE	Return from Interrupt				
Syntax:	<code>[label] RETFIE s</code>				
Operands:	$s \in [0,1]$				
Operation:	$(TOS) \rightarrow PC$, $1 \rightarrow GIE/GIEH$ or $PEIE/GIEL$, if $s = 1$ $(WS) \rightarrow W$, $(STATUSS) \rightarrow STATUS$, $(BSRS) \rightarrow BSR$, PCLATU, PCLATH are unchanged.				
Status Affected:	GIE/GIEH, PEIE/GIEL, STATUS reg.				
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0000</td><td>0000</td><td>0001</td><td>000s</td></tr></table>	0000	0000	0001	000s
0000	0000	0001	000s		
Description:	Return from Interrupt. Stack is popped and Top of Stack (TOS) is loaded into the PC. Interrupts are enabled by setting the either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).				

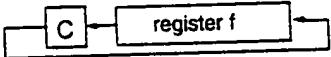
Words: 1
Cycles: 2
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example: `RETFIE Fast`

After Interrupt

PC	=	TOS
W	=	WS
BSR	=	BSRS
STATUS	=	STATUSS
GIE/GIEH, PEIE/GIEL	=	1

RLCF	Rotate Left f through Carry		
Syntax:	[label] RLCF f,d,a		
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]		
Operation:	(f<n>) → dest<n+1>, (f<7>) → C, (C) → dest<0>		
Status Affected:	C,N,Z		
Encoding:	0011 01da ffff ffff		
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
			
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLCF REG, 0, 0

Before Instruction
 REG = 1110 0110
 C = 0

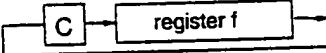
After Instruction
 REG = 1110 0110
 W = 1100 1100
 C = 1

RLNCF	Rotate Left f (no carry)		
Syntax:	[label] RLNCF f,d,a		
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]		
Operation:	(f<n>) → dest<n+1>, (f<7>) → dest<0>		
Status Affected:	N,Z		
Encoding:	0100 01da ffff ffff		
Description:	The contents of register 'f' are rotated one bit to the left. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).		
			
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLNCF REG, 1, 0

Before Instruction
 REG = 1010 1011

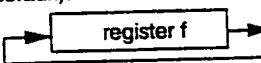
After Instruction
 REG = 0101 0111

RRCF	Rotate Right f through Carry		
Syntax:	[label] RRCF f,d,a.		
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]		
Operation:	(f<n>) → dest<n-1>, (f<0>) → C, (C) → dest<7>		
Status Affect d:	C,N,Z		
Encoding:	0011 00da ffff ffff		
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).		
			
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RRCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
W = 0111 0011
C = 0

RRNCF	Rotate Right f (no carry)		
Syntax:	[label] RRNCF f,d,a.		
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]		
Operation:	(f<n>) → dest<n-1>, (f<0>) → dest<7>		
Status Affected:	N,Z		
Encoding:	0100 00da ffff ffff		
Description:	The contents of register 'f' are rotated one bit to the right. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).		
			
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111

After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111

After Instruction
W = 1110 1011
REG = 1101 0111

SETF	Set f			
Syntax:	[label] SETF f,a			
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$			
Operation:	FFh $\rightarrow f$			
Status Affected:	None			
Encoding:	0110 100a ffff ffff			
Description:	The contents of the specified register are set to FFh. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1 Q2 Q3 Q4			
	Decode	Read register \ddagger	Process Data	Write register \ddagger

Example: SETF REG, 1

Before Instruction
REG = 0x5A

After Instruction
REG = 0xFF

SLEEP	Enter SLEEP mode			
Syntax:	[label] SLEEP			
Operands:	None			
Operation:	00h \rightarrow WDT, 0 \rightarrow WDT postscaler, 1 \rightarrow TO, 0 \rightarrow PD			
Status Affected:	TO, PD			
Encoding:	0000 0000 0000 0011			
Description:	The power-down status bit (PD) is cleared. The time-out status bit (TO) is set. Watchdog Timer and its postscaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1 Q2 Q3 Q4			
	Decode	No operation	Process Data	Go to sleep

Example: SLEEP

Before Instruction

TO = ?
PD = ?

After Instruction

TO = 1 \dagger
PD = 0

\dagger If WDT causes wake-up, this bit is cleared

<u>SUBFWB</u>		<u>Subtract f from W with borrow</u>							
Syntax:	[label]. SUBFWB f,d,a.								
Operands:	0 ≤ f ≤ 255 d ∈ {0,1} a ∈ {0,1}								
Operation:	(W) - (f) - (C) → dest								
Status Affected:	N,OV,C,DC,Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0101</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table>		0101	01da	ffff	ffff			
0101	01da	ffff	ffff						
Description:	<p>Subtract register 'f' and carry flag (borrow method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register f</td><td>Process Data</td><td>Write to destination</td></tr> </table>		Q1	Q2	Q3	Q4	Decode	Read register f	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register f	Process Data	Write to destination						

<u>SUBFWB</u>	
<u>Example 1:</u>	SUBFWB REG, 1, 0
Before Instruction	
REG	= 3
W	= 2
C	= 1
After Instruction	
REG	= FF
W	= 2
C	= 0
Z	= 0
N	= 1 ; result is negative
<u>Example 2:</u>	SUBFWB REG, 0, 0
Before Instruction	
REG	= 2
W	= 5
C	= 1
After Instruction	
REG	= 2
W	= 3
C	= 1
Z	= 0
N	= 0 ; result is positive
<u>Example 3:</u>	SUBFWB REG, 1, 0
Before Instruction	
REG	= 1
W	= 2
C	= 0
After Instruction	
REG	= 0
W	= 2
C	= 1
Z	= 1
N	= 0 ; result is zero

SUBLW	Subtract W from literal
Syntax:	[label] SUBLW k.
Operands:	0 ≤ k ≤ 255
Operation:	$k - (W) \rightarrow W$
Status Affected:	N,OV, C, DC, Z
Encoding:	0000 1000 kkkk kkkk
Description:	W is subtracted from the eight bit literal 'k'. The result is placed in W.
Words:	1
Cycles:	1
Q Cycle Activity:	
	Q1 Q2 Q3 Q4
	Decode Read literal 'k' Process Data Write to W

Example 1: SUBLW 0x02

Before Instruction

W = 1
C = ?

After Instruction

W = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2: SUBLW 0x02

Before Instruction

W = 2
C = ?

After Instruction

W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: SUBLW 0x02

Before Instruction

W = 3
C = ?

After Instruction

W = FF ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF	Subtract W from f
Syntax:	[label] SUBWF f,d,a.
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	$(f) - (W) \rightarrow \text{dest}$
Status Affected:	N,OV, C, DC, Z
Encoding:	0101 11da ffff ffff
Description:	Subtract W from register f (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register f (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write to destination

Example 1: SUBWF REG, 1, 0

Before Instruction

REG = 3
W = 2
C = ?

After Instruction

REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: SUBWF REG, 0, 0

Before Instruction

REG = 2
W = 2
C = ?

After Instruction

REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: SUBWF REG, 1, 0

Before Instruction

REG = 1
W = 2
C = ?

After Instruction

REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

SUBWFB Subtract W from f with Borrow

Syntax: [label] SUBWFB f,d,a

Op rands: $0 \leq f \leq 255$

$d \in [0,1]$

$a \in [0,1]$

Operation: $(f) - (W) - (\bar{C}) \rightarrow \text{dest}$

Status Affected: N,OV, C, DC, Z

Encoding:

0101	10da	ffff	ffff
------	------	------	------

Description: Subtract W and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (d fault).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

SUBWFB

Example 1: SUBWFB REG, 1, 0

Before Instruction

REG = 0x19 (0001 1001)
W = 0x0D (0000 1101)
C = 1

After Instruction

REG = 0x0C (0000 1011)
W = 0x0D (0000 1101)
C = 1
Z = 0
N = 0 ; result is positive

Example 2: SUBWFB REG, 0, 0

Before Instruction

REG = 0x1B (0001 1011)
W = 0x1A (0001 1010)
C = 0

After Instruction

REG = 0x1B (0001 1011)
W = 0x00
C = 1
Z = 1 ; result is zero
N = 0

Example 3: SUBWFB REG, 1, 0

Before Instruction

REG = 0x03 (0000 0011)
W = 0x0E (0000 1101)
C = 1

After Instruction

REG = 0xF5 (1111 0100) [2's comp]
W = 0x0E (0000 1101)
C = 0
Z = 0
N = 1 ; result is negative

SWAPF Swap f

Syntax:	[label] SWAPF f,d,a								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>								
Status Affected:	None								
Encoding:	<table border="1"><tr><td>0011</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0011	10da	ffff	ffff				
0011	10da	ffff	ffff						
Description:	The upper and lower nibbles of register f are exchanged. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed in register T (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Read register T</td><td>Process Data</td><td>Write to destination</td></tr></tbody></table>	Q1	Q2	Q3	Q4	Decode	Read register T	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register T	Process Data	Write to destination						

Example: SWAPF REG, 1, 0

Before Instruction

REG = 0x53

After Instruction

REG = 0x35

TBLRD	Table Read				
Syntax:	[label] TBLRD (*; *+; *-; +*)				
Operands:	None				
Operation:	<p>if TBLRD *, (Prog Mem (TBLPTR)) → TABLAT; TBLPTR - No Change;</p> <p>if TBLRD *+, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) +1 → TBLPTR;</p> <p>if TBLRD *-, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) -1 → TBLPTR;</p> <p>if TBLRD +*, (TBLPTR) +1 → TBLPTR; (Prog Mem (TBLPTR)) → TABLAT;</p>				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>10nn nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table>	0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*
0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*		
Description:	<p>There are four options with a TBLRD instruction to determine what happens to the 21-bit Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately, and the contents of the program memory location pointed to by the TBLPTR are loaded into the 8-bit Table Latch (TABLAT). The Lsb of the TBLPTR selects which byte of the program memory location will be read. If Lsb = 1, the high byte will be loaded into the TABLAT. If Lsb = 0, the low byte will be loaded into the TABLAT.</p>				
Words:	1				
Cycles:	2				
Q Cycle Activity:					

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (OE goes low) TABLAT updated

TBLRD	Table Read
Example1:	TBLRD *+ ;
Before Instruction	
TABLAT	= 0x55
TBLPTR	= 0x00A356
MEMORY(0x00A356)	= 0x34
After Instruction	
TABLAT	= 0x34
TBLPTR	= 0x00A357
Example2:	TBLRD +* ;
Before Instruction	
TABLAT	= 0xAA
TBLPTR	= 0x01A357
MEMORY(0x01A357)	= 0x12
MEMORY(0x01A358)	= 0x34
After Instruction	
TABLAT	= 0x34
TBLPTR	= 0x01A358

TBLWT	Table Write	TBLWT	Table Write												
Syntax:	[label] TBLWT (*; *+; *-; +)	Cycles:	2 (many if long write is to on-chip EPROM program memory)												
Operands:	None	Q Cycle Activity:													
Operation:	<p>if TBLWT *, $(TABLAT) \rightarrow \text{Prog Mem}(TBLPTR);$ $TBLPTR - \text{No Change};$</p> <p>if TBLWT *+, $(TABLAT) \rightarrow \text{Prog Mem}(TBLPTR);$ $(TBLPTR) +1 \rightarrow TBLPTR;$</p> <p>if TBLWT *-, $(TABLAT) \rightarrow \text{Prog Mem}(TBLPTR);$ $(TBLPTR) -1 \rightarrow TBLPTR;$</p> <p>if TBLWT +*, $(TBLPTR) +1 \rightarrow TBLPTR;$ $(TABLAT) \rightarrow \text{Prog Mem}(TBLPTR);$</p>	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation (Table Pointer on Address bus)</td><td>No operation</td><td>No operation (Table Latch on Address bus, WR goes low)</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	No operation	No operation	No operation	No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, WR goes low)	
Q1	Q2	Q3	Q4												
Decode	No operation	No operation	No operation												
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, WR goes low)												
Status Affected:	None	Example 1: TBLWT *+;													
Encoding:	<table border="1"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>11nn</td> </tr> <tr> <td></td> <td></td> <td></td> <td>nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table>	0000	0000	0000	11nn				nn=0 * =1 *+ =2 *- =3 +*	Before Instruction	TABLAT = 0x55 TBLPTR = 0x00A356 MEMORY(0x00A356) = 0xFF				
0000	0000	0000	11nn												
			nn=0 * =1 *+ =2 *- =3 +*												
Description:	<p>There are four options with a TBLWT instruction. These options determine what happens to the Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately.</p> <p>The contents of Table Latch (TABLAT) are written to the program memory location pointed to by TBLPTR.</p> <p>If TBLPTR points to an external program memory location, then the instruction executes in two cycles.</p> <p>Since the TABLAT is only one byte wide, a multiple of two TBLWT instructions must be executed to program internal memory locations. For example, if the device is defined to program one word at time, an internal memory location is programmed in the following manner:</p> <ol style="list-style-type: none"> 1) Set TBLPTR to an even byte 2) Write low byte to TABLAT 3) Execute TBLWT *+ (2-cycle) 4) Write high byte to TABLAT 5) Execute TBLWT *+ (long write) <p>A long write to an internal EPROM location is terminated when an interrupt is received. The post increment TBLWT instruction is the only TBLWT instruction that is recommended for writes to internal memory. (Writes to internal EPROM are only available on devices with 64 or more pins.)</p>	After Instructions (table write completion)	TABLAT = 0x55 TBLPTR = 0x00A357 MEMORY(0x00A356) = 0x55												
Words:	1	Example 2: TBLWT +*;													
		Before Instruction	TABLAT = 0x34 TBLPTR = 0x01389A MEMORY(0x01389A) = 0xFF MEMORY(0x01389B) = 0xFF												
		After Instruction (table write completion)	TABLAT = 0x34 TBLPTR = 0x01389B MEMORY(0x01389A) = 0xFF MEMORY(0x01389B) = 0x34												

RAP		Debugger Subroutine Call					
Syntax:	[label] TRAP						
Operands:	None						
Operation:	(PC) + 2 → TOS, 000028h → PC<20:1>						
Status Affected:	INBUG						
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0000</td><td>0000</td><td>1110</td><td>0000</td></tr></table>	0000	0000	1110	0000		
0000	0000	1110	0000				
Description:	Debugger Trap to 00028h. First, return address (PC+ 2) is pushed onto the return stack. Then the 20-bit value '000028h' is loaded into PC<20:1>. The INBUG status bit is set. TRAP is a two-cycle instruction.						
Words:	1						
Cycles:	2						
Q Cycle Activity:							
	Q1	Q2	Q3	Q4			
	Decod	Push PC to stack	No operation	Write 000028h to PC			
	No operation	No operation	No operation	No operation			

Example: HERE TRAP

Before Instruction

PC = Address (HERE)

After Instruction

PC = 000028h
TOS = Address (HERE + 2)
INBUG = 1

TRET		Trap Return from Subroutine					
Syntax:	[label] TRET						
Operands:	None						
Operation:	(TOS) → PC PCLATU, PCLATH are unchanged						
Status Affected:	INBUG						
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0000</td><td>0000</td><td>1110</td><td>0001</td></tr></table>	0000	0000	1110	0001		
0000	0000	1110	0001				
Description:	Return from debugger trap. The stack is popped and the top of the stack (TOS) is loaded into the program counter. The INBUG status bit is cleared.						
Words:	1						
Cycles:	2						
Q Cycle Activity:							
	Q1	Q2	Q3	Q4			
	Decode	No operation	No operation	pop PC from stack			
	No operation	No operation	No operation	No operation			

Example: TRET

After Interrupt

PC = TOS
INBUG = 0

TSTFSZ	Test f, skip if 0
Syntax:	[label] TSTFSZ f,a
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	skip if $f = 0$
Status Affected:	None
Encoding:	0110 011a ffff ffff
Description:	If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    TSTFSZ CNT, 1
NZERO  :
ZERO   :
```

Before Instruction

PC = Address(HERE)

After Instruction

```
If CNT    = 0x00,
PC      = Address (ZERO)
If CNT    = 0x00.
PC      = Address (NZERO)
```

XORLW Exclusiv OR literal with W

Syntax:	[label] XORLW k.			
Operands:	$0 \leq k \leq 255$			
Operation:	$(W) .XOR. k \rightarrow W$			
Status Affected:	N,Z			
Encoding:	0000 1010 kkkk kkkk			
Description:	The contents of W are XOR'ed with the 8-bit literal 'k'. The result is placed in W.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
Q1	Q2	Q3	Q4	
Decode	Read literal 'k'	Process Data	Write to W	

Example: XORLW 0xAF

Before Instruction

W = 0xB5

After Instruction

W = 0x1A

XORWF Exclusiv OR W with f

Syntax: [label]. XORWF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$ Operation: (W) .XOR. (f) \rightarrow dest

Status Affected: N,Z

Encoding:

0001	10da	ffff	ffff
------	------	------	------

Description: Exclusive OR the contents of W with register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in the register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register f	Process Data	Write to destination

Example: XORWF REG, 1, 0

Before Instruction

REG = 0xAF
W = 0xB5

After Instruction

REG = 0x1A
W = 0xB5